1969

A Comparison of Some Numerical Integration Programs

J. Casaletto

M. Pickett

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
69-037

A COMPARISON OF SOME NUMERICAL INTEGRATION PROGRAMS

J. Casaletto, M. Pickett and J. Rice

CSD TR 37

June 1969

A COMPARISON OF SOME NUMERICAL INTEGRATION PROGRAMS

1. INTRODUCTION. This report summarizes in a concise form the results of the testing a number of programs for numerical integration. Fourteen programs (all in FORTRAN) were examined and seven were tested. The testing of one (RIEMAN), of these was not completed since it was seen to be significantly inferior to all others.

The aim was to judge these programs for general purpose use and applicability to a wide variety of functions. Further all the programs tested required only that the integrands, the limits and an accuracy indicator be given. Thus each program is a candidate for the automatic numerical integration program of a computing center, subroutine library or high level problem oriented mathematical system.

The six programs completely tested are all good quality programs though there is still significant variation in their quality. They are all superior to routines that one would write on the basis of the material in current numerical analysis texts, even if this were done with care. It is of some interest that none of the methods in these texts is useful for this problem.

2. NUMERICAL INTEGRATORS COLLECTED.

Name	Most Immediate Source
GAUSS	- Davis & Rabinowitz - Numerical Integration
SUM	"
PS	"
FILONT	"
AVINT	"
RIEMAN	"
QUAD	"
SIMP	

ROMBRG-(POLY) C. de Boor & Tamblyn, Purdue University

TRAP : "

QUAD : " (hereafter cited as QUADS)

SIMPSN : "

ROMBRG : "

SQUANK : J. Lyness, Argonne National Labs

3. DISPOSITION AND REMARKS

GAUSS and SUM are utility routines in which the user must provide significant input and thus are not self-contained numerical integrators. These routines are not considered further in this report.

P5 was designed for integration over hyper-rectangles excluding dimension 1.

FILONT was designed for integrating functions of the form f(x)cosax and f(x)sinax, but a could not be set to zero in the first case and setting a to k≠ in either case would introduce an error.

AVINT was designed for integrating tabulated functions.

Because these three routines are so specific, they could not be compared to others; so they were deleted from further consideration.

ROMBRG-(POLY) and TRAP did not compile on the CDC 6500 and were also deleted.

The RIEMAN routine allows the user to specify an $N \geq 2$, and then uses an N-point Riemann sum adaptively. This routine was tested with N varying from 2 to 20. Initial observations show these RIEMAN routines to be very unreliable and extremely inefficient.*

The routine QUADS requires the user to specify the use of a 4 point or a 6 point Gaussian quadrature rule; so that we refer to QUADS 4 and QUADS 6 when appropriate.

---

*In this report, efficiency and economy are synonymous and are measured by the number of function evaluations a routine uses in estimating the value of an integral.

SIMP and SIMPSN are almost identical. In the few cases where they behave differently, SIMPSN is the better of the two.

The routines included in the comparison are thus

RIEMAN

QUAD

SIMP

QUADS

SIMPSN

ROMBRG

SQUANK

4. THE TEST PROCEDURE

The integrators tested were put to the task of integrating 50 functions with requested relative accuracies from 10^{-1} to 10^{-8}. This results in 400 test cases for a comparison of these integrators. The test functions, their intervals of integration, and the correct numerical value of the integral, exact to the number of digits displayed, are presented in the Appendix. (These functions are refered to as #8, #47, etc. in the summary which follows).

This report is a summary of the observations made on the basis of these 400 tests. Each integrator was limited to a maximum of 10,000 function evaluations. If a routine attempted to exceed this limitation for any request, it was terminated and is listed as an Overflow. However, the routine QUADS has a built-in restriction of 2400 for the 4 point formula and 4800 for the 6 point formula; thus, in the summary, there are no overflows listed and such a condition is listed as a Quit.

If a routine returned a value which was not within the relative (L_4) accuracy requested, it is termed a Failure.

If a routine does not show, through the eight different accuracy requests, that it can integrate a certain function, it is termed a Total Failure on that function. However, if indications are that the routine can integrate the function, yet it failed to produce the requested accuracy in at least one of the cases beyond the case of a relative accuracy request of .01, then it is termed as Unreliable for that function. In some instances, a routine will have a low accuracy failure; that is, it can integrate it, but it stops short because of an accidental type convergence when a low relative accuracy is requested. Such a condition is noted separately.

5. SUMMARY OF TEST RESULTS

QUAD:

12 Failures- accuracy not met

12 Overflowed (10,000 function evaluations)

4 Low accuracy problems #29(.1), 31(.1,.01),34(.1)

Overflows occurred on:

#31(10^{-5},10^{-6},10^{-7},10^{-8})

#39(10^{-5},10^{-6},10^{-7},10^{-8})

#48(10^{-5},10^{-6},10^{-7},10^{-8})

Total failure on #30

QUADS 4:

21 Failures

15 Quit (has own limit of 2400 function evaluations)

2 Low accuracy problems #29(.1,.01)

Quit on:

#30(10^{-2},...,10^{-8})

#34(10^{-2},10^{-3},10^{-4},10^{-8})

#50(.1,10^{-2},10^{-3},10^{-7})

Unreliable on #26, 31, 40

Total failure on #30,39,48

QUADS 6:

7 Failures

14 Quit

1 Low Accuracy problem #29(.1)

Quit on:

#30(10^{-2},...,10^{-8}),#34(10^{-2},10^{-3},10^{-4},10^{-8}),#50(10^{-3},10^{-7},10^{-8})

No cases of unreliability

Total failure on #30,48

SIMP:

5 Failures

12 Overflows

5 Low accuracy problems #30(.1,.01),31(.01),34(.1,.01)

Overflows occurred on #30(10^{-6},10^{-7},10^{-8}),31(10^{-7},10^{-8}),50(10^{-1},...,10^{-8})

No cases of unreliability

Total failure on #50

SIMPSN:

6 Failures

5 Overflows

6 Low accuracy problems #30(.1,.01),31(.01),34(.1,.01),50(.01)

Overflows occurred on #30(10^{-6},10^{-7},10^{-8}),31(10^{-7},10^{-8})

No cases of unreliability

No Total failures

SQUANK:

29 Failures

7 Overflows

8 Low accuracy problems #29(10^{-1},10^{-2},10^{-3}),31(10^{-1},...,10^{-4}),60(10^{-5})

Overflows occurred on 29(10^{-8}),31(10^{-8}),34(10^{-4},...,10^{-8})

Unreliable on #48

Total failures on #30,#45,#46

ROMBRG:

19 Failures

0 Overflows

8 Low Accuracy problems #26(.1,.01),29(.1),30(.1,.01),31(.1),34(.1),50(.01)

No overflows occurred

No cases of unreliability

Total failures on #45,46.

RIEMAN N:

This routine is unreliable and works too hard.

The following table gives the minimum number of function evaluations for each program (the first row). Then the average number of function evaluations used is tabulated for eight levels of accuracy and the six programs (including two cases for QUADS). These averages are for functions which are extremely smooth and which present no problems to a numerical integration program.

AVERAGE NUMBER OF FUNCTION EVALUATIONS
FOR "EXTRA NICE" FUNCTIONS #1→21,24,28, and 35 - $\int_0^1 f(x)dx$

	QUAD	QUADS-4	QUADS6	SIMP	SIMPSN	SQUANK	ROMBRG
MIN	9	12	24	19	19	9	5
10^{-1}	9	12	24	19	19	9	5
10^{-2}	9	12	24	19	19	9	7.2
10^{-3}	11.2	12	24	19	19	9.2	8.6
10^{-4}	21.3	12	24	25	25	13.3	13
10^{-5}	27.3	12	24	44	44	19.1	16.8
10^{-6}	40.3	12	24	55.6	55.6	34	23.5
10^{-7}	60.3	12	24	116	116	63.3	33
10^{-8}	66	12.5	25	153.5	153.5	102.5	39.4

6. COMMENTS ON INDIVIDUAL ROUTINES

ROMBRG:

(i) Fails for step functions (45,46)

(ii) Works too hard on functions (36,...,42)

(iii) Works a little more than other routines on functions 26 and 29; yet does very well on 31 and 34.

(iv) Low accuracy problems might be eliminated by modifying a low accuracy request i.e. if a relative accuracy of .1 is requested, this could be changed to .05 or even .01.

This routine is very economical. With a modification on low accuracy requests and a suggestion, to the user, to break up step functions into their continuous interval, ROMBRG becomes very reliable.

SQUANK:

(i) Fails on step functions (45,46)

(ii) Works a little harder on functions (34,49)

(iii) Works well on functions (36,37,38,39,40,41)

(iv) Works excellent on functions (42,43,44)

This routine is a modification of SIMP, it sacrifices a little reliability for a greater economy. Of 50 functions tested, SIMP could not integrate 1 while SQUANK could not integrate 3 (2 of these are noted in (i) above). Other than these failures, SIMP outdoes SQUANK only on function 49. In general, SQUANK is twice as economical. If the user were to break up step functions appropriately, then both routines would have the same number of total failures and hence equal reliability on these test functions. Also SQUANK has a few more low accuracy problems than SIMP, but SIMP pays for it.

QUAD:

(i) Fails only on function #30

(ii) Otherwise, reliable

(iii) Not economical

(iv) In a few cases, it is the best but this does not make up for its general loss in economy

(v) Works hard on functions (31,36,37,38,39,40,41,42,48)

QUADS: (4 point, 6 point)

(i) Very little low accuracy problems

(ii) The most economical at higher accuracy requests

(iii) 4 pt. unreliable on oscillating integrands

(iv) 6 pt. fairly reliable

Although these routines internally limit their number of function evaluations to 2400 and 4800, respectively, there is no indication that they

ever work too excessively. In those cases in which the routines quit at higher accuracy requests, a review of the lower accuracy requests shows them to be working efficiently in relation to the other integrators.

SIMP and SIMPSN:

(i) A few low accuracy problems

(ii) The most reliable

(iii) Very inefficient, sacrifices economy for reliability

In general these routines work much harder than all the other routines except for the RIEMAN routine. SIMPSN is better than SIMP in integrating functions #37,38,39,40,45,46,48 and 50; on all the other functions, they are identical.

APPENDIX

I	$F_I(x)$	a	b	$\int_a^b F_I(x)dx$
1	1	0	1	1.00000000
2	x-2	0	1	-1.50000000
3	x^2-2x+3	0	1	2.33333335
4	x^3-2x^2+3x-4	0	1	-2.91666667
5	$xF_4(x)+5$	0	1	3.70000000
6	$xF_5(x)-6$	0	1	-4.31666667
7	$xF_6(x)+7$	0	1	5.07619048
8	$xF_7(x)-8$	0	1	-5.71071429
9	$xF_8(x)+9$	0	1	6.45634921
10	$xF_9(x)-10$	0	1	-7.10198413
11	$xF_{10}(x)+11$	0	1	7.83852814
12	$xF_{11}(x)-12$	0	1	-8.49173882
13	$xF_{12}(x)+13$	0	1	9.22187257
14	$xF_{13}(x)-14$	0	1	-9.88057776
15	$xF_{14}(x)+15$	0	1	10.60594961
16	$xF_{15}(x)-16$	0	1	-11.26882146
17	$xF_{16}(x)+17$	0	1	11.99051684
18	$xF_{17}(x)-18$	0	1	-12.65665666
19	$xF_{18}(x)+19$	0	1	13.37542806
20	$xF_{19}(x)-20$	0	1	-14.04419947
21	e^x	0	1	1.71828183
22	SIN(πx)	0	1	.63661977
23	COS x	0	1	.84147098
24	$x/(e^x-1)$	0	1	.77750463
25	$1/(1+x^2)$	0	1	.78539816

I	$F_I(x)$	a	b	$\int_a^b F_I(x)dx$
26	$2/(2+SIN(10\pi x))$	0	1	1.15470054
27	$1/(1+x^4)$	0	1	.866972987
28	$1/(1+e^x)$	0	1	.37988549
29	$xSIN(30x)COSx$	0	2π	-.20967247
30	$xSIN(30x)COS(50x)$	0	2π	.11780972
31	$xSIN(30x)/\sqrt{1-x^2/4\pi^2}$	0	2π	-2.543260
32	$\frac{23}{25}COSHx-COSx$	-1	1	.4794282
33	$1/(x^4 + x^2 + .9)$	-1	1	1.582233
34	$(100\pi^2 - x^2)1/2SINx$	0	10π	26.42050
35	$1/(1+x)$	0	1	.69314718
36	$x^{1/2}$	0	1	.66666667
37	$x^{1/4}$	0	1	.80000000
38	$x^{1/8}$	0	1	.88888889
39	$x^{1/16}$	0	1	.94117647
40	$\sqrt{\lvert x^2 - .25\rvert}$	0	1	.46474250
41	$x^{3/2}$	0	1	.40000000
42	$\sqrt{\lvert x^2 - .25\rvert^3}$	0	1	.14867162
43	$x^{5/2}$	0	1	.28571429
44	$\sqrt{\lvert x^2 - .25\rvert^5}$	0	1	.065514765
45	$[10x]$	0	1	4.500000000
46	$\begin{cases} x, & 0\leq x\leq .333 \\ x+1, & .333\leq x\leq .667 \\ x+2, & .667\leq x\leq 1 \end{cases}$	0	1	1.500000000
47	$\begin{cases} 0, & .49<x<.50 \\ -1000(x^2 - x), & \text{otherwise} \end{cases}$	0	1	164.16700

I	$F_I(x)$	a	b	$\int_a^b F_I(x)dx$
48	$\begin{cases} 1/(x+2) & ,0\le x\le 0.2 \\ 0 & ,0.2<x\le 1 \end{cases}$	0	1	.30685282 ✓
49	$10^4(x-.1)(x-.11)(x-.12)(x-.13)$	0	1	1089.25266...
50	SIN(100πx)	0	1	0.00000000 ✓

```
      FUNCTION F51(X)
C     PIECEWISE, ALMOST CONTINUOUS, CANNOT SEE BREAK POINTS
      IF(X.GT.EXP(-X))GOTO10
      IF(TAN(X).GT.1.01*X)GOTO30
      F51=SQRT(X)-.236*SQRT(ABS(.5-X*(16.+30.*X)))
      RETURN
   10 IF(TAN(X)-COS(X).GT..5)GOTO20
      FS1=(X+ALOG(ABS(X)))*(X-.855)
      RETURN
   20 FS1=SIN(X)*(1.+SIN(X))-.5*(2.+COS(X))
      RETURN
   30 FS1=(1.+X**4-2.*X**X*COSH(X))*(X*COS(X)-1.01*SIN(X)*
     *(COS(X))**2*(.96*X*X+1.)*(COSH(X)-COS(X))
      RETURN
      END
```